runlinc Communication Project 2: The Binary System (STEMSEL Version)

Contents

INTRODUCTION **1**

A SIMPLE APPLICATION: BINARY REPRESENTATION **4**

PART A: DESIGN THE CIRCUIT ON RUNLINC 5
PART B: BUILD THE CIRCUIT 5
PART C: PROGRAM THE CIRCUIT 5
SUMMARY 7

Introduction

What is the binary system?

The binary system is when a system consists of a sequence of only two states. And it can be represented in the STEM field as a numerical system which uses only 2 numbers, 0 and 1. In more technical terms, it is a numerical system which uses base 2. An example of a binary number is 11110100. A digit in binary is called a binary digit which is often shortened to bit. So, the example 11110100 has 8 digits, which is the same as saying it's 8 bits long. When you have 8 bits altogether, it is known as a byte. Also, when reading a binary number, read each digit individually.

Why do we use the binary system?

In electrical and computer fields, hardware has at least 2 primary methods to process electrical signals. One is analog and the other is digital. Analog uses the smooth and continuous form of an electrical signal to function. Whereas, in the digital method, any information conveyed by an analog electric signal will be broken down into a sequence of binary states of electricity being on or off. The on and off means we can use binary mathematics to design hardware for function-specific outputs.

The particularity of the digital system means we can store binary data by keeping the on and off state for a data in a particular module for later use and then use a binary arithmetic module to manipulate the stored data. It is more difficult to store information in an analog system and more likely not to be able to use the stored information for further processing. This is a critical principle which allows us to create logic circuits and therefore microchips and computers.

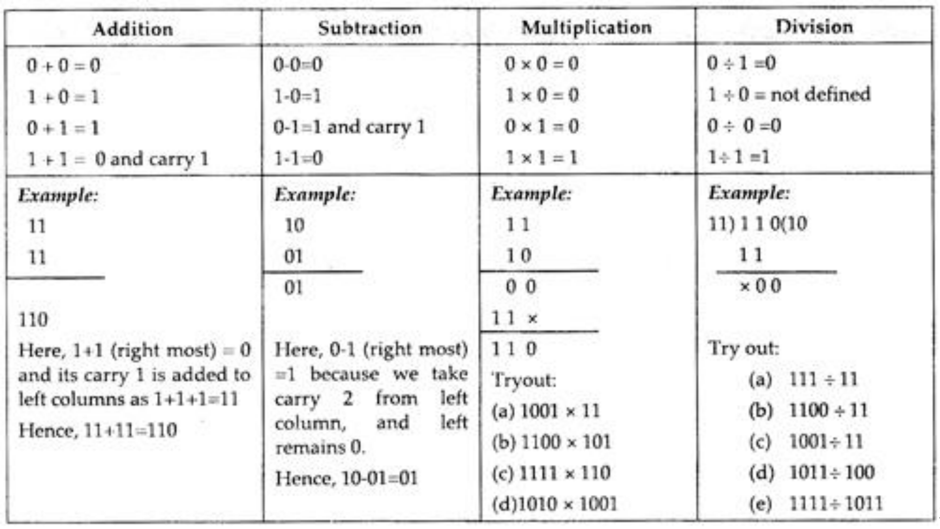
Additionally, since an analog system uses the original form of an electrical signal, electromagnetic effects from other devices can easily affect the analog system as it will affect the signal’s amplitude. Whereas in the digital system, one can be certain a signal is either on or off at a certain amplitude threshold, this means that electromagnetic effects from other devices are less likely to affect the digital system. Therefore, combining this knowledge with the ability for further processing, many newer electronics will use the digital method and thus, it is necessary to understand the binary system.

However, it should be noted that digital signals cannot convert a continuous (infinite) signal into a binary form without taking up too many digits to store the signal. This means a binary number with fewer digits (bits) is less precise, and it should also be noted that the digital form can also be converted back to analogue form, but it will be affected by the number of bits that the digital signal contains.

How does binary numeral system work?

The binary numeral system works similar to the commonly used decimal system (0 to 9). To count in binary, in a digit, we count from 0 to 1. At 1, if we need to count one more, we start from 0 but carry 1 to the left. If we are counting in reverse, we count from 1 to 0, and to count further, we borrow 1 from the left. You can use usual arithmetic methods on binary numbers the same way, see the following figure. Additionally, you can convert between binary and decimal, the following link goes into the details of this:

https://www.electronics-tutorials.ws/binary/bin_2.html



Addition	Subtraction	Multiplication	Division
0 + 0 = 0 1 + 0 = 1 0 + 1 = 1 1 + 1 = 0 and carry 1	0-0=0 1-0=1 0-1=1 and carry 1 1-1=0	0 × 0 = 0 1 × 0 = 0 0 × 1 = 0 1 × 1 = 1	0 ÷ 1 =0 1 ÷ 0 = not defined 0 ÷ 0 =0 1÷ 1 =1
Example: 11 11 ——— 110 Here, 1+1 (right most) = 0 and its carry 1 is added to left columns as 1+1+1=11 Hence, 11+11=110	*Example:* 10 01 ——— 01 Here, 0-1 (right most) =1 because we take carry 2 from left column, and left remains 0. Hence, 10-01=01	*Example:* 1 1 1 0 ——— 0 0 1 1 × ——— 1 1 0 Tryout: (a) 1001 × 11 (b) 1100 × 101 (c) 1111 × 110 (d)1010 × 1001	*Example:* 11) 1 1 0(10 1 1 ——— × 0 0 Try out: (a) 111 ÷ 11 (b) 1100 ÷ 11 (c) 1001÷ 11 (d) 1011÷ 100 (e) 1111÷ 1011

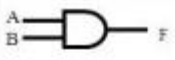
[1]

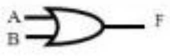
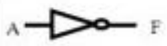
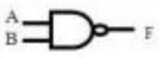
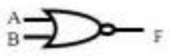
Figure 1 Binary Arithmetic Methods.

[1] https://www.sciencehq.com/wp-content/uploads/Binary-Arithmetic.jpg

In mathematics, there is a field called Boolean algebra where binary states can be manipulated using logic. It is a simple but powerful mathematic model that helps us designing hardware. By using Boolean algebra, we can take some particular input and then logically state the output. Then apply this to an electric circuit, which once we apply this concept, is called a logic circuit. In this field, 1 and 0 can be thought as true and false, or, on and off, respectively. The following figure will show the fundamental logic needed for Boolean algebra. Watch the following link to understand more about Boolean algebra: https://www.youtube.com/watch?v=gI-qXk7XojA

Name	Graphic Symbol	Boolean Algebra	Truth Table

AND		F = A . B
		Or
		F = AB
			A B F
			0 0 0
			0 1 0
			1 0 0
			1 1 1

OR		$F = A + B$	A B F 0 0 0 0 1 1 1 0 1 1 1 1
NOT		$F = \overline{A}$	B F 0 1 1 0
NAND		$F = \overline{A . B}$ Or $F = \overline{AB}$	A B F 0 0 0 0 1 1 1 0 1 1 1 1
NOR		$F = \overline{A + B}$	A B F 0 0 0 0 1 0 1 0 0 1 1 1

[2]

Figure 2 Fundamental Boolean Logic

Summary

We hope that this brief introduction has allowed you to understand what the binary system is, why we use it in the STEM field, especially in the electronics and computing, and that you learned how the binary system can be manipulated using arithmetic and Boolean algebra.

runlinc Background

runlinc is a web page inside a Wi-Fi chip. The programming is done inside the browsers compared to programming inside a chip. The runlinc web page inside the Wi-Fi chip will command the microchips to do sensing, control, data logging Internet of Things (IoT). It can predict and command

[2] https://maecellesther.files.wordpress.com/2017/10/boolean.jpg?w=487&h=365

A Simple Application: Binary Representation

Problem

Think of the 'trapped miners' application in the telecommunication introduction. We used a programmable microchip on the surface which was connected to the light in the cave visible to the miners, which we turned on and off to communicate the code to the miners. But what if instead of this situation, we need to convey other types of information to the trapped miner? In the new situation, the trapped miner will need both alphabet and numeral information to disable the explosives.

Ideas

How can we use binary to represent more complex information? Computers use binary sequences to function, so how can we see Latin characters? Think of morse code as in the previous trapped miner application. How did people communicate with one another with morse code? How should we signify if a letter or numeral is being sent?

Plan

We will build upon the previous trapped miner setup. We will first need to decipher if the following character for the combination is a number or an alphabet. We should use the first two digits of the binary sequence to decipher if the rest of the sequence is a number or a letter. The first digit will always be on, to allow the miner to know that the sequence has started. The second digit will be off to signify that the rest of the sequence is a number. Whereas, when the second is on, the rest of the sequence is a letter.

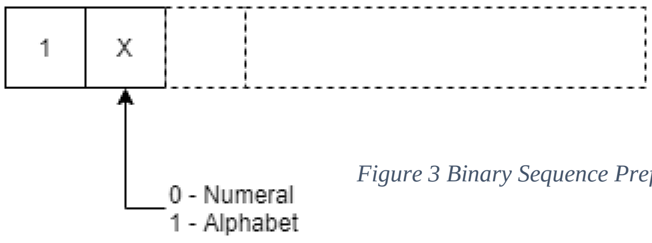


Figure 3 Binary Sequence Prefix

But here is the problem, if we turn off the light entirely during the off digit, the miners will not know if the sequence is for the next code and also will have a hard time, finding the position of the bits in the sequence. Therefore, we will set that a long flash is an on and a very short flash is an off. And we will be using decimal to binary and binary to decimal conversion, in order to shorten the on and off flashing time.

If you still haven't thought of a way to encode alphabets, no worries. It's simple to encode the alphabet into binary. We will use a method which computers store and use alphabets. We simply encode the respective alphabets in alphabetical order with their respective numerical number.

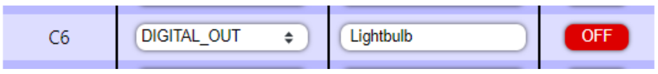
Part A: Design the Circuit on runlinc

Note: Refer to runlinc Wi-Fi Setup Guide document to connect to runlinc

Use the left side of the runlinc web page to construct an input/output (I/O).

In our circuit design,

- ❖ C6 -> DIGITAL_OUT: Lightbulb



C6 | DIGITAL_OUT | Lightbulb | OFF

Figure 4 Expected I/O Configuration.

Part B: Build the Circuit

Use the runlinc I/O to connect the hardware. Remember that turning the screws clockwise will close the clamps and turning the screws anticlockwise will open them. All black wires should go in the negative (-) terminal, red wires go in the positive (+) terminal, and white wires go in the terminal we designated in the runlinc web page port.

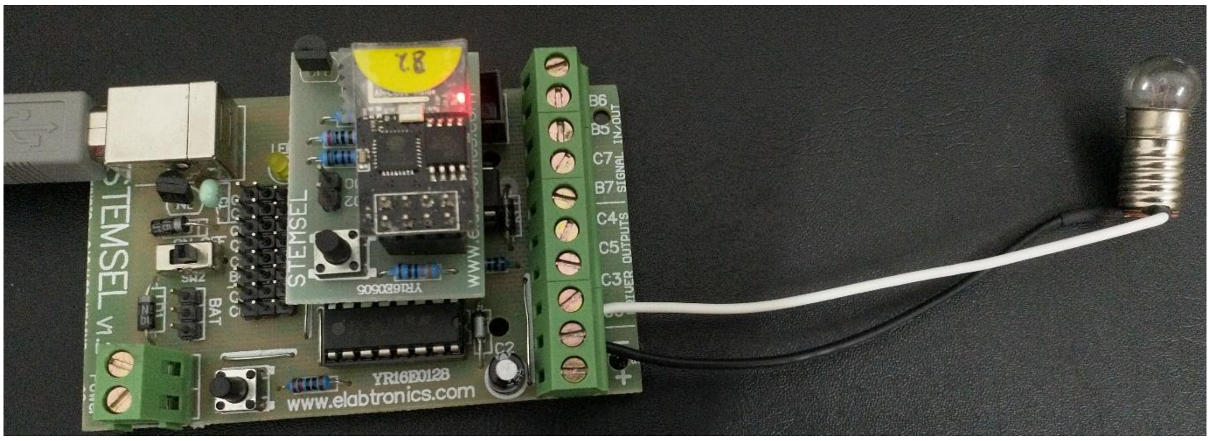


Figure 5 Circuit connection on STEMSEL.

<u>Wiring Instructions</u>

- ❖ Connect All White Wires to their Respective Pins
  - ➢ Lightbulb -> C6
- ❖ Connect All Black Wires to negative (-) terminal.

Part C: Program the Circuit

<u>HTML</u>

We will first set up a simple HTML page to show what the miners will see, such that we can confirm it is the message we want to send. We will use a circle to mimic the lightbulb. Let’s label the circle with a heading, so we remember that the circle represents the lightbulb.

```
<h1>This is what the miner will see.<h1>
<svg height="100" width="100">
  <circle id="circle" cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="white" />
</svg>
```

JavaScript

We will write several functions in JavaScript which will help us sending information to the miners.

1. We will write an on function which has long turn on time and a short turn off time.

```
async function on(){
   turnOn( Lightbulb );
   document.getElementById("circle").style.fill="gold";
   await mSec( 1500 );
   turnOff( Lightbulb );
   document.getElementById("circle").style.fill="white";
   await mSec( 500 );
   return Promise.resolve();
}
```

2. Then an off function that has the opposite light behaviour.

```
async function off(){
   turnOn( Lightbulb );
   document.getElementById("circle").style.fill="gold";
   await mSec( 500 );
   turnOff( Lightbulb );
   document.getElementById("circle").style.fill="white";
   await mSec( 1500 );
   return Promise.resolve();
}
```

3. Then we will write a blink function that will utilize the on and off function. For the blink function, we will first initialize with an on. Then check if this blink message is a number or a letter. Then we will turn the value into a binary message and send the message to the miners by utilizing decimal to binary conversion. Numerals will utilize 4 bits because 4 bits go up to 16 unique values whereas 3 bits go up to 8 unique values, so i = 3. Letters will utilize 5 bits since it can provide 32 unique values.

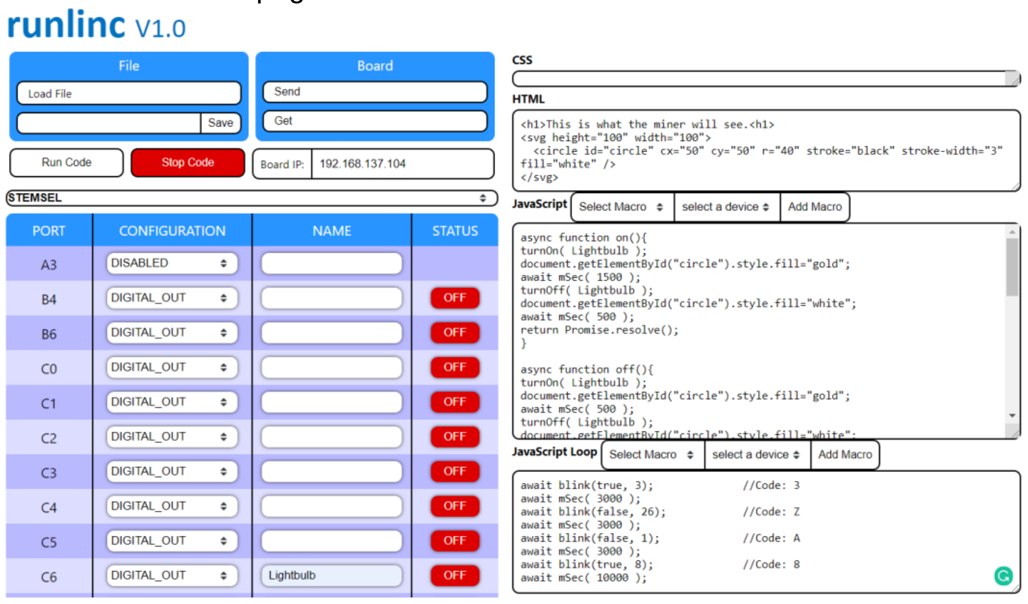
```
async function blink(isNumber, value){
   await on();
   //Check if it is number of alphabet.
   if( isNumber ){
      await on();
   }else{
      await off();
   }
   let v = value;
   let i = 3;              // Set i = 3 or 4 depending on if it is a number or an alphabet.
   if( !isNumber ){
      i = 4;
   }
   // Decimal to binary conversion readable by the miners.
   for( i ; i>=0; i--){
      if( (v>>i) & 1 == 1){
         await on();
      }else{
         await off();
      }
   }
return Promise.resolve();
}
```

<u>JavaScript Loop</u>

The JavaScript Loop will send the message to the miners using our blink function. Between each blink message will need a 3 second delay, and the last message will have a 10 second delay afterwards.

```
await blink(true, 3);              //Code: 3
await mSec( 3000 );
await blink(false, 26);              //Code: Z
await mSec( 3000 );
await blink(false, 1);              //Code: A
await mSec( 3000 );
await blink(true, 8);               //Code: 8
await mSec( 10000 );
```

Expected runlinc and web page:



PORT	CONFIGURATION	NAME	STATUS
A3	DISABLED		
B4	DIGITAL_OUT		OFF
B6	DIGITAL_OUT		OFF
C0	DIGITAL_OUT		OFF
C1	DIGITAL_OUT		OFF
C2	DIGITAL_OUT		OFF
C3	DIGITAL_OUT		OFF
C4	DIGITAL_OUT		OFF
C5	DIGITAL_OUT		OFF
C6	DIGITAL_OUT	Lightbulb	OFF

HTML

```
<h1>This is what the miner will see.<h1>
<svg height="100" width="100">
  <circle id="circle" cx="50" cy="50" r="40" stroke="black" stroke-width="3"
fill="white" />
</svg>
```

JavaScript

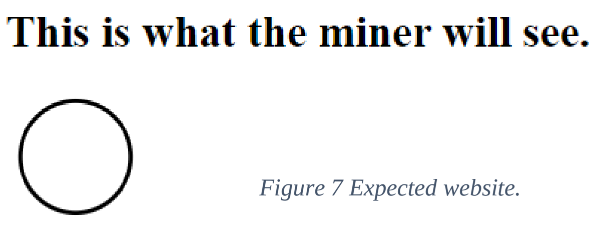
```
async function on(){
turnOn( Lightbulb );
document.getElementById("circle").style.fill="gold";
await mSec( 1500 );
turnOff( Lightbulb );
document.getElementById("circle").style.fill="white";
await mSec( 500 );
return Promise.resolve();
}

async function off(){
turnOn( Lightbulb );
document.getElementById("circle").style.fill="gold";
await mSec( 500 );
turnOff( Lightbulb );
document.getElementById("circle").style.fill="white";
```

JavaScript Loop

```
await blink(true, 3);             //Code: 3
await mSec( 3000 );
await blink(false, 26);           //Code: Z
await mSec( 3000 );
await blink(false, 1);            //Code: A
await mSec( 3000 );
await blink(true, 8);             //Code: 8
await mSec( 10000 );
```

Figure 6 Expected runlinc



This is what the miner will see.

Figure 7 Expected website.

Summary

This further application of the miner problem should introduce you to a simple utilization of binary numbers in electronics. Instead of counting numbers as we did in the first miners project, you can use binary to send more information with less space. Additionally, you would have learned how numerals and letters can be encoded inside the computer. To further your understanding of how this system works, you should research ASCII encoding and then Unicode encoding. These will help you in further manipulation of information in computing and programming.